Prelude

Purpose

To develop an algorithm for extraction of license plate numbers from still photos of stationary cars.

Methods

We divided this project into two sections- Image Processing and Support Vector Machine (SVM) Training:

- We take picture of a still car (from either front or back), compress and crop out to speed up the process, localize the plate position and extract the letters into individual binary matrices.
- We take each of the matrix generated in the previous process and use the SVM algorithm to distinguish which letter exactly it is.

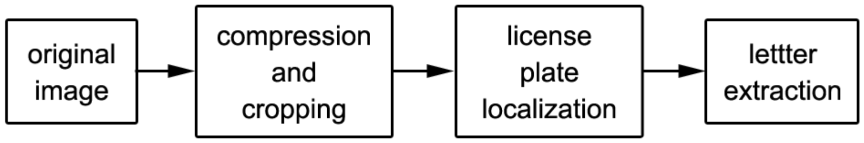
Applications

This algorithm can be integrated to the related vehicle identification applications such as:

1. Parking lot registration
2. Traffic violation tracking
3. Vehicle surveillance

Image Processing - License Plate Localization and Letters Extraction
ELEC 301 Group Project: LiPE How to find the license plate letters and numbers in an image of a car.

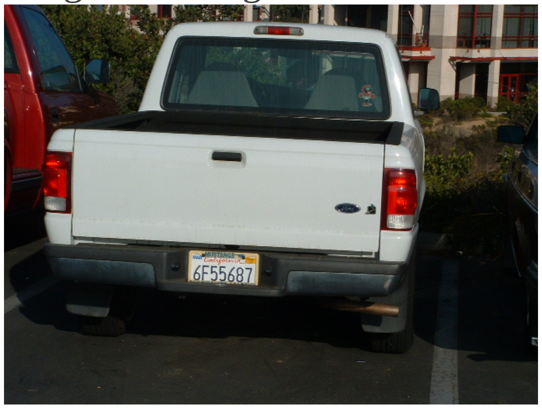
The general method we used for extracting the license plate letters out of a picture was:



Overall method for finding license plate letters/numbers.

1. **Compression and Cropping:** decreases the size of the photo and and blacks out areas that definitely do not contain license plate.
2. **License Plate Localization:** determines the location of the plate in the photo
3. **Letter Extraction:** searches within the plate for the plate letters/numbers and copies them out of the photo.

Following is an explanation of the individual sections. Included are also images showing the effects of each section on the following picture:



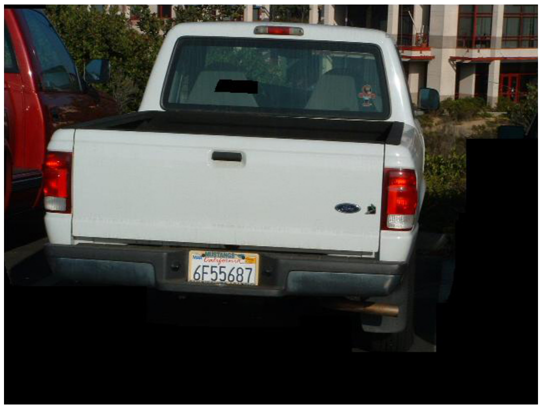
Original image.

Compression and Cropping

In order to decrease image processing time, we first compress all pictures to a standard size and black out all areas that are definitely not license plates. The size we chose was 640px by 480px, the smallest size at which we could still reasonably read the license plates.

To determine which areas were definitely not license plates, we realized that the typical Texas plate contained red, blue, and white as major colors. Therefore, we focused on these two colors, making our cropping algorithm:

1. Separate the JPEG picture into its three layers of red, green, and blue.
2. Consider the area around a blue pixel, and black it out if the density of red is lower than a certain threshold value.



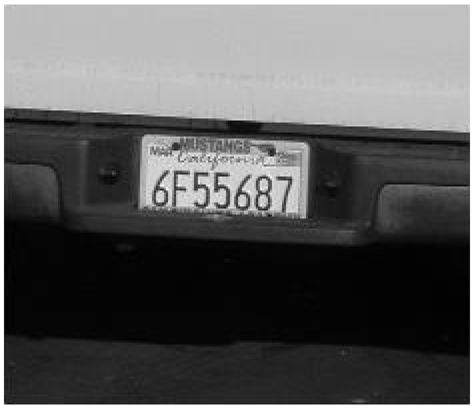
Compressed and cropped image (Note the black areas around the right and bottom of the photo).

License Plate Localization

Once we determined which areas possibly contained license plates, we looked in those areas for the plates themselves. We determined that most plates contain dark letters on light backgrounds and so looked for areas of high contrast.

Our final algorithm looked as follows:

1. Turn the cropped photo into black-and-white for easier differentiation between dark and light spots.
2. Filter the image to remove noise (single-pixel white spots).
3. Locate the license plate position by scanning the photo vertically. We expect a row running through the license plate row to have a maximum number of individual dark spots, or "clusters." Therefore, we find and store the two rows in the image with that contain the most clusters.
4. To find the horizontal position of the plate, scan the picture horizontally by moving a square window from left to right and counting the number of clusters inside. The final position of the license plate is square that contains the greatest number of clusters. If any two squares contain the same number of clusters, the two are merged together.



Close-up of the license plate as determined by the algorithm.

Letter Extraction

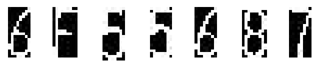
To find the letters on a license plate, we first determined some identifying characteristics:

1. Usually, and always on Texas plates, the letters of the plate are dark on a light background.
2. The letters are uniform in height
3. The letters all occur in approximately the same area.
4. There are usually between 3 and 7 letters on a plate.

These characteristics give us the form for our letter extracting algorithm.

1. From the plate-locating algorithm, we have a small 200px by 200px image that contains the car's license plate.
2. We convert the image into grayscale for easier processing.
3. We locate all the dark (low intensity) spots in the picture that are surrounded by light (high intensity) spots. We determine the separation between these dark spots and give each individual spot a label.
4. Compare the sizes of the spots and look for about six that have the same height. These six spots are the letters on the plate.
5. Save the pixels that make up the letters into their individual matrices.

We tried this algorithm and found that it worked for all images for which the plate-locating algorithm returned an image containing the entire plate.



Letters extracted from the photo.

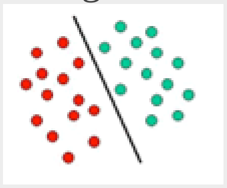
SVM Train

The Math and Algorithm

For digit recognition, we use Support Vector Machine (SVM) as a learning machine to perform multi-class classification.

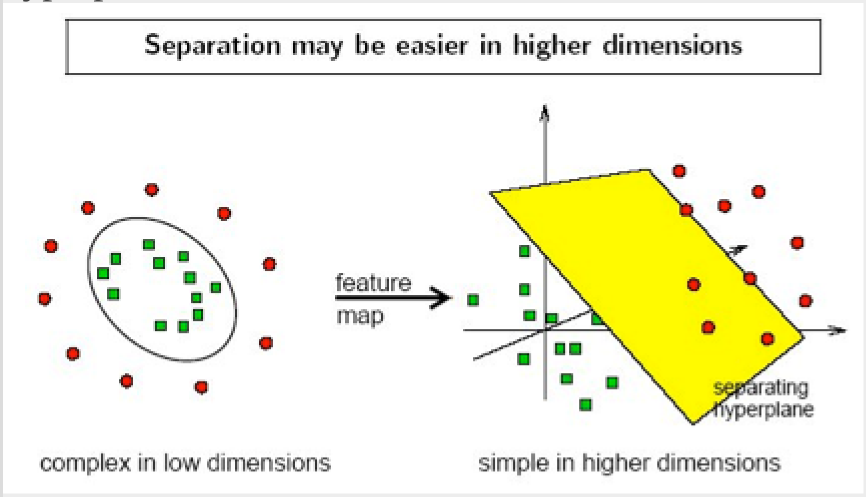
The way SVM works is to map vectors into an N-dimensional space and use an (N-1)-dimensional hyperplane as a decision plane to classify data. The task of SVM modeling is to find the optimal hyperplane that separates different class membership.

Example:
Let's take a look at a simple schematic example where every object either belongs to GREEN or RED.



Picture from: Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines, 2001.

SVM finds the line defining the boundary between the two types. Then, it can classify a new object by looking at on which side of the line it falls. However, it is unlikely that we can always have a linear dividing boundary. Rather than fitting nonlinear curves to the data, we can map each object into a different space via a kernel function where a linear dividing hyperplane is feasible.


Separation may be easier in higher dimensions
feature map
separating hyperplane
complex in low dimensions
simple in higher dimensions

Hyperplane classifying the two cases (Picture from: Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines, 2001.)

The concept of the kernel mapping function is so powerful that SVM can perform separation with very complex boundaries. The kernel function we use in this project is radial basis function (RBF).

SVM Working Principles

- Vectorize each instance into an array of features (attributes).
- Model with training data to find optimal dividing hyperplane with maximal margin.
- Use SVM to map all the objects into a different space via a kernel function (see Figure 3 for examples).

- Classify new object according to its position with respect to hyperplane.
- Errors in training are allowed while the goal of training is to maximize the margin and minimize errors. Namely, find the solution to optimization problem in Figure 4, where x is the attribute, y is the object label, ξ is the error and φ is the mapping function.

- linear: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$.
- polynomial: $K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i{}^T \mathbf{x}_j + r)^d,\ \gamma > 0$.
- radial basis function (RBF): $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2),\ \gamma > 0$.
- sigmoid: $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \mathbf{x}_i{}^T \mathbf{x}_j + r)$.

A few examples on the kernel functions; for our case we choose the radial basis function (RBF)

$$\begin{aligned} \min_{\mathbf{w}, b, \boldsymbol{\xi}} \quad & \frac{1}{2}\mathbf{w}^T \mathbf{w} + C \sum_{i=1}^{l} \xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T \phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0. \end{aligned}$$

Find a hyperplane with the maximized margin space to split the two classes. (Equation from: Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines, 2001.)

Methods and Running Routines

1. Collect the matrices obtained from the Image Processing section and label each of the instances.
2. Select a reasonable amount as the training set, and the rest as the testing set, with a balanced choice for each of the instances.
3. Feed the training set into the SVM-train process to generate a model. This calculation takes time, but it only needs to be done once.
4. Now we're ready to make predictions to a given license plate! An input of labeled data will give us the accuracy of this algorithm; an unlabeled instance can also be fed in to see the prediction.

Note: The SVM library we used is available at: http://www.csie.ntu.edu.tw/~cjlin/libsvm

Conclusions

Results

We were able to successfully locate the license plate in about 70% of our sample pictures, and of the characters we extracted, we were able to recognize them with about 72.7% accuracy (619 sets of training data).

Future Improvements

Compression and Cropping

Since we were targeting Texas plates, which only have red, blue, and white colors, we were able to black out many parts of the images by wiping out all green regions. In the future, however, we would like to be able to recognize plates not from Texas that might have green components. Therefore, we should find a criteria for finding the plates other than color.

Letter Recognition

Acquiring State Pattern and Convention Attributes
In many license plates, it is difficult to tell the difference between a zero and an O, even for a human. Therefore, for the purposes of this project, zeros and Os were considered the same. However, in many states, it is actually possible to tell the difference because the license plate has a set pattern (e.g., 2 letters, 2 numbers, 2 letters). In the future, we could identify what state the plate comes from and then make use of this knowledge to get more accuracy in letter recognition.

Multi-class Support Vector Machine
In addition, one of the characteristics of SVM is that is solves a two-class problem. In order to get around this, for our project, we used a one-against-the-rest approach. This meant that we basically used the SVM machine to answer the question, "Is this a __?" 35 (A-Z, 1-9) times for each unknown letter/digit. In the future, we will want to look at more efficient and accurate

methods. One of the possible improvement can be found in the work by *T.-K. Huang, R. C. Weng, and C.-J. Lin.* *Generalized Bradley-Terry Models and Multi-class Probability Estimates. Journal of Machine Learning Research*

Automated Training Set Generation and Extraction Efficiency
Finally, currently, any digit that we feed into the SVM will register as something; we have no way of telling whether the image is in fact a letter/digit. In the future, we would like to train the machine to be able to tell characters from non-characters. This will allow less rigorous (and time-consuming) computation in the image-processing section and give our algorithm greater flexibility.

References

Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

Special Thanks

Thanks to:

- Dr. Aswin Sankaranarayanan, our mentor
- Dr. Richard Baraniuk, the ELEC 301 instructor
- Dr. Fatih Porikli (MERL), for providing us with a license plate dataset
- Drew Bryant and Brian Bue for technical advising